Efficient Data Aggregation with Secure Bloom Filter in Wireless Sensor Networks

Zhijun Li[1] and Guang Gong[2]
[1] CORBU branch
Cisco Systems Canada Co.
Ottawa, Ontario, K2K3E8 Canada
[2]Department of Electrical and Computer Engineering
University of Waterloo, Waterloo, Ontario, Canada
Emails: ansli@cisco.com, ggong@uwaterloo.ca

Abstract

Designing secure data aggregation schemes, which are critical to many sensor network applications, imposes interesting and formidable challenges. In this paper, we propose a succinct and practical secure aggregation protocol by combining HMAC (associated with a cryptographic hash function) with Bloom filter, which then is defined as secure Bloom filter. Unlike most previous approaches, which are aimed to provide security mechanisms for ordinary aggregation operations, our proposal firstly is an effective aggregation protocol, suitable for a specific but popular class of aggregation in wireless sensor networks. Benefiting from secure Bloom filter, the protocol, without any unrealistic assumptions, fulfills the fundamental security objective of preventing outside adversaries and compromised inside nodes from harming the overall network result. We systematically analyze the protocol performance and run extensive simulations on different network scenarios for evaluation. The simulation results demonstrate that the proposed protocol presents remarkable performance on security, communication cost, and degree of node energy consumption balance.

Keywords. Bloom filter, secure Bloom filter, secure data aggregation, Wireless sensor networks.

1 Introduction

In typical wireless sensor networks (WSNs), hundreds and thousands of low-cost sensor nodes scatter within a surveillance area, receive commands from a base station, perform designat-

ed detection tasks accordingly, and collaboratively transmit results back to the base station in a multi-hop, infrastructure-less communication pattern. In many cases, instead of forwarding every individual message to the base station, sensor network protocols support data aggregation—the operation by intermediate nodes that combines many messages and sends out aggregated results. As a matter of fact, from the very beginning of wireless sensor networks development [1, 2, 3, 4, 5], it has already been widely accepted that data aggregation plays a critical role in the practicability and appealing of WSNs. Due to infeasibility of recharging nodes batteries in most circumstances, energy becomes the most valuable resource for sensor nodes. Among all nodes operations, data transmission consumes energy the most [2, 5]. Intuitively, data aggregation during message transmission is an effective method to preserve sensor nodes precious energy. Moreover, in the absence of data aggregation, sensor nodes near the base station will suffer from heavy message transmission overhead, and then die of power exhaustion much sooner than other nodes, breaking down the whole network's functionality. As a result, data aggregation attracts a great deal of attention and many a data aggregation scheme has been proposed in recent years. Interested readers may refer to [3, 4] for systematic surveys on this topic.

When sensor nodes are deployed in a hostile environment, security measurements should be taken into consideration for network protocols. In general, attacks to wireless sensor networks not only come from outside adversaries, but also can be conducted by compromised, previously legitimate nodes. This kind of attack model, along with the fact that sensor nodes are very resource-constrained, imposes formidable challenges for designing secure data aggregation. When purely cryptographic mechanisms are used for securing data aggregation, homomorphic primitives might be the only suitable candidates, and then many literatures follow this track. However, as we will argue later, the application of homomorphic primitives on data aggregation is highly limited. For other kinds of secure data aggregation approaches, sophisticated protocols are designed to detect malicious behaviors/nodes. Some of those mechanisms rely on unrealistic assumptions, while other involve heavy communication overhead, which conflicts with the very intention of data aggregation and makes it hard to evaluate their applicability. There are some well-designed secure data aggregation schemes, which understandably have different trade-offs and are suitable for particular operations.

In this paper, we propose a secure data aggregation protocol which fits a specific but popular class of aggregations in wireless sensor networks. Unlike most previous approaches, which are aimed to provide security mechanisms for ordinary aggregation operations, our proposal in the first place is an effective and succinct aggregation protocol, which is equipped with built-in security mechanisms, fulfilling the fundamental security purpose of preventing compromised nodes as well as outside adversaries from harming network aggregated results. The security technique backing up our protocol is to combine HMAC, a conventional and provably secure cryptographic primitive, with the space-efficient data structure Bloom filter [6]. The resulting data structure is then defined as secure Bloom filter, and its security implications are thoroughly addressed in this paper. We use it to construct a secure aggregation protocol

for sensor networks. The theoretical analysis and extensive simulation results demonstrate that the proposed protocol presents remarkable performance on security, communication cost, and energy consumption balance degree among sensor nodes.

The rest of this paper is organized as follows. First, previous work on secure data aggregation is discussed in Section 2. Then we state application scenarios, and define network and adversary models in Section 3. Afterward, we present secure Bloom filter in Section 4. Section 5 details our proposed secure aggregation protocol and corresponding analysis. The simulation design and results are elaborated in Section 6. Finally, we conclude our work in Section 7.

2 Previous Work

Homomorphic primitives, besides standard cryptographic functionalities, allow users without secret key to legitimately perform acceptable algebraic operations on protected data block. Since aggregation is essentially some operation, it is intuitive to use homomorphic primitives for securing data aggregation, and a number of approaches [7, 8, 9] use homomorphic encryption for this application. Generally, it is very difficult to design secure symmetric homomorphic block encryption, whereas stream ciphers naturally support homomorphic exclusive-OR operations, which is exactly utilized in [8]. In contrast, public-key homomorphic encryption is an interesting topic, and there exist several relatively practical public-key homomorphic cryptosystems, such as Unpadded RSA, El-Gamal, Goldwasser-Micali, Benaloh and Paillier [10], though all of them only support limited operations on ciphertexts. In 2009, Gentry [11], for the first time, presented a fully homomorphic encryption scheme, which outstandingly allows arbitrary operations on ciphertext. Even though the only two fully homomorphic encryption schemes [11, 12] by now cannot provide competitive performance for most applications, practical fully homomorphic encryption is expected to appear eventually.

Unfortunately, sole public-key homomorphic encryption does not suffice for secure data aggregation in sensor networks, because then anyone can maliciously insert or manipulate results. Even for the stream homomorphic encryption in [8], its security demands message integrity mechanisms as well, and their solution to that issue is essentially a homomorphic MAC. The topic of using homomorphic MAC and homomorphic hashing in secure data aggregation is systematically discussed in [13]. Applying those kinds of homomorphic primitives to data aggregation either involves cumbersome key management problems, or incurs considerable communication costs. Consequently, their applicabilities are highly limited.

Hu and Evans [14] described a secure hop-by-hop data aggregation scheme, and later Jadia and Muthuria [15] extended their scheme. However, both schemes are only capable of preventing a single inside malicious node at an appreciable communication cost, rendering them impractical. Yang *et al.* [16] presented a secure hop-by-hop data aggregation protocol for sensor networks named SDAP, using the principles of divide-and-conquer and commit-and-

attest to design complicated algorithms which cause significant transmission overhead, and may cancel off all communication benefits from data aggregation.

Przydatek, Song, and Perrig [17] proposed secure information aggregation (SIA) to identify forged aggregation values from malicious nodes. In the SIA scheme, a special node named aggregator computes an aggregation result over raw data together with a commitment to the data based on a Merkle-hash tree and sends them back to a remote user, which later challenges the aggregator to verify the aggregation. Later Chan, Perrig, and Song [18] built a hierarchical data aggregation on the aggregate-commit-prove framework in [17], but extended their single aggregator model to a fully distributed setting. Frikken and Dougherty [19] further improved the Chan-Perrig-Song scheme. Moreover, Chan and Perrig [20] derived several security primitives from this kind of algorithms.

3 Background

3.1 Application Scenarios

As a typical data aggregation application scenario of our proposal, a wireless sensor network is employed in a hostile environment to detect a number of *predefined, basic* events with only two states: ON and OFF. For most of time, these events stay OFF: they did not occur. When sensor nodes detect an event taking place, they notify the base station about their discovery. Since there are many nodes simultaneously detecting same events, data aggregation is desired to reduce communication costs and to balance nodes energy consumption. The base station may be tolerant of a little inaccuracy on final aggregated results.

Even though the scenario looks like relatively simplistic, it is fairly popular for wireless sensor network applications. For instance, an intrusion-detection sensor network [21] successfully triggers an alarm for a perimeter intrusion event. Moreover, sophisticated tasks can be decomposed into many basic events. In such cases, sensor nodes are responsible for simple event detection, which significantly reduces their production expenditure, while the base station performs advanced data analysis. Other similar application scenarios are voting and counting mechanisms for sensor networks. For example, in many secure sensor network protocols, malicious nodes are determined by majority results from nodes voting; and under many circumstances, the base station may need to frequently count the number of nodes that match some standards.

3.2 Network and Adversary Models

Homogeneous Sensor Network: We consider an ordinary, homogeneous sensor network that consists of n low-cost sensor nodes which are resource-constrained in terms of computation, communication, storage, and power supply. Those sensor nodes are especially sensitive to

energy consumption and thus data aggregation is desired. As a rule, there exists a central, powerful base station that is responsible for collecting sensor network results.

Adversary Model: Sensor nodes are not equipped with tamper-proof hardware, and the adversary is capable of compromising and fully controlling arbitrary number of sensor nodes, but we assume that the base station is immune to all physical attacks and it is trustworthy. Moreover, the adversary can eavesdrop and alter any messages from integrity nodes. In a word, the adversary is granted the full-scale attack capacity against the sensor network except the base station.

Multi-hop Data Aggregation: In our network model, sensor nodes may be densely deployed, or nodes may only have a few neighbors. Intuitively, the network size n should not be small and most of communications between nodes and the base station are multi-hop; otherwise, there is no point for introduction of data aggregation. Our protocol does not rely on specific data aggregation modes; any effective aggregation organization, no matter it is cluster-, chain-, tree-, or grid-based [3], suffices for the proposed protocol. We do not address the problem how to construct a specific aggregation organization, because it is quite irrelevant to our proposal.

Terminology: For the purpose of convenient demonstrations, the following terminology is used. There are two kinds of roles for sensor nodes: a *contributor* that detects events and generates *raw messaes*, and an *aggregator* that aggregates all messages that it received plus possibly its own raw messages [1] and then forwards *aggregated messages*. Messages are aggregated though the network; and the base station eventually retrieves a number of aggregated messages, which are defined as *reports*.

Base Station Authenticated Broadcast: It is helpful for final data analysis if the base station is aware of all nodes' positions; but it is not mandatory for the proposed protocol. We assume that there is a secure broadcast message authentication scheme available for the base station, which is pretty natural for securing sensor networks. Otherwise, it is infeasible for the base station to issue reliable, integrity commands to sensor nodes and manage the sensor network. This simple requirement can be easily satisfied for wireless sensor networks. We may use the popular, lightweight, symmetric-key hash-chain-based μTESLA authentication protocol [22] for this purpose. Alternatively, all nodes are preloaded with the base station's public key, and then messages from the base station are signed and verified through classical signature schemes. For simplicity, we ignore specific choices on broadcast message authentication.

Simplistic Key Predistribution Requirement: Every node shares a distinct long-term key with the base station. Those keys are pre-distributed into sensor nodes prior to deployment. No pairwise keys between sensor nodes or any other kinds of keys are demanded. Sensor nodes are equipped with a cryptographic hash function. Those constitute all security primitive requirements of the proposed protocol. As a general method to reduce its key storage, the base station may employ a pseudo-random generator (PRG) along with one main key, which generates node's key using a node's ID and the main key as input of PRG.

[1] In such a case, the aggregator is also a contributor.

3.3 Security Objectives

Data confidentiality is not a concern for the application scenarios at which our proposal is targeted. For simple event detection networks, if an adversary is physically close to sensor nodes, he can easily sense those simple events on his own. Certainly, some extent of message privacy is appreciated. Barely given a message and without any other knowledge, should an adversary not tell which nodes have contributed to the message and what events are reported.

Generally speaking, the main security goal of secure aggregation is to strictly restrain adversary's influence only on those compromised nodes. First, even with collusion of all other sensor nodes, an adversary cannot fake a specific node's event report with more than a low, theoretically-limited probability. Second, other than compromised nodes' input on an event report, the adversary is only able to affect accumulative results in a trivial way. Otherwise, the scheme should detect the abnormality and discard related results. If intermediate nodes manipulate or drop integrity nodes' event-report messages, the secure aggregation scheme should be able to effectively detect the attacks.

3.4 Native Solution without Data Aggregation

There exists a straightforward, native solution to fulfilling the aforementioned security objectives, which does not engage data aggregation and then incurs heavy communication cost. When node u detects an event e, it generates a raw message composed of $\mathsf{Mac}(K_u, e)$ and sends to the base station, where K_u is the symmetric key shared by node u and the base station, and Mac denotes a tag by a standard deterministic MAC. When the base station receives a message, it may try all combinations to figure out which node claims which event. In order to prevent message-dropping by malicious nodes, the base station broadcasts all messages it received through the broadcast authentication mechanism. If a node does not obtain the authenticated acknowledgment of its raw message in a reasonable time slot, it will notify the base station about a potential of the attack.

It may seem tempting to use aggregate MAC [23], which aggregates multiple MAC tags by different contributors into a single tag that can be verified by the base station, to reduce the communication payload of the native scheme. Unfortunately, in such an application, an aggregate tag has to list all contributors and associated events such that the base station can verify the tag. Consequently, the size of an aggregated message is still proportional to the number of contributors and events. Moreover, it violates the message privacy objective.

4 Secure Bloom Filter

The technical core of our proposal is a variant of Bloom filter, for which we coin the term *secure Bloom filter*.

4.1 Foundation: Bloom Filter

Bloom filter, conceived by Bloom [6], is a space-efficient probabilistic data structure that succinctly represents a set in order to support membership queries. Due to its distinguished space advantages and excellent distributed properties, Bloom filter has been widely used in numerous areas, such as web cache sharing [24] and distributed storage system [25].

4.1.1 Typical Implementation of Bloom Filter

Typically, a Bloom filter is implemented as a bit-array of m bits associated with h different hash functions, each of which maps an element to one of the m array positions in a uniformly random manner. All bits in an initial Bloom filter are set to 0, standing for an empty set. To insert an element e into a set represented by a Bloom filter BF, h array positions are calculated by hash functions on e and the bits at those positions in BF are set to 1. Correspondingly, when it is required to check the membership of an element v within the Bloom filter BF, supplying v to hash functions outputs h array positions; if any of the bits at the h positions is 0, then element v does not belong to the set; otherwise, the element is claimed to be a member of the set.

4.1.2 Properties of Bloom Filter

It is easy to see that there is no *false negative* in the Bloom filter membership verification—an element which tests negative within a Bloom filter definitely is not a legitimate member of the set. On the other hand, Bloom filter may yield *false positive*: a member outside the set passing the membership verification on the Bloom filter. The probability of a false positive for an element not in the set, or the *false positive probability*, can be calculated in a straightforward manner.

Let ζ be the probability of a bit being 0 in a Bloom filter, the false positive probability is then

$$P_{\mathrm{FP}} = (1-\zeta)^h \quad . \tag{1}$$

After inserting t elements in a Bloom filter, assume that the hash functions outputs are independently uniformly distributed, we have

$$\zeta = \left(1 - \frac{1}{m}\right)^{ht} \approx e^{-ht/m} \quad ;$$

the false positive probability is therefore

$$P_{\mathrm{FP}} = \left(1 - \left(1 - \frac{1}{m}\right)^{ht}\right)^h \approx \left(1 - e^{-ht/m}\right)^h \quad . \tag{2}$$

The right hand side is minimized when

$$h = \frac{m}{t} \ln 2 \approx 0.6931 \frac{m}{t} \ .$$

In such a case, $\zeta = 0.5$, and the false positive probability is $2^{-h} \approx (0.6185)^{m/t}$. Hence, the optimal results are achieved when each bit of the Bloom filter is 0 with probability 1/2.

In a typical parameters determination procedure of Bloom filter, users specify desired false positive probability P_{FP} and predicted inserted element number t, then h and m can be calculated via

$$h = \lceil -\log_2 P_{\mathrm{FP}} \rceil$$

and

$$m = \lceil ht / \ln 2 \rceil \ .$$

4.2 Specification of Secure Bloom Filter

Suppose that the Bloom filter is used in a distributed environment and each element is associated with a specific user claiming an event. It is easy to see that the original Bloom filter data structure does not support a necessary security property—A malicious user or an adversary which is allowed to manipulate the data can forge other users' inputs. We introduce the *secure Bloom filter* to prevent this attack. Formally, it is defined as follows.

Definition 1 (Secure Bloom Filter)**.** *Suppose that every user is allocated with a random, secret key. A secure Bloom filter is a data structure that not only maintains all properties of the original Bloom filter, but also guarantees the infeasibility of an adversary constructing a valid input associated with a user claiming an event, without the knowledge of the user's key. The probability of a successful forging is defined as the advantage of an adversary against the secure Bloom filter, and is denoted by* Adv*.*

Secure Bloom filter can been constructed by deploying a cryptographic hash function, through HMAC, to substitute a family of hash functions for the Bloom filter.

4.2.1 Definition of a Standard HMAC

We use the cryptographic hash function by means of HMAC because it is provably secure [26] and has been standardized [27]. Let $\mathsf{H}(*)$ denote a cryptographic hash function. According to RFC 2104 [27], to authenticate a message e with a secret symmetric key K, $\mathsf{HMac}(K, e)$ is defined by

$$\mathsf{HMac}(K, e) = \mathsf{H}\left((K \oplus \mathrm{opad}) \parallel \mathsf{H}((K \oplus \mathrm{ipad}) \parallel e)\right),$$

where $\parallel$ denotes concatenation, $\oplus$ denotes exclusive-OR (XOR), opad is the one-block-size outer padding (0x5c5c...5c), and ipad is the one-block-size inner padding (0x3636...36).

4.2.2 Construction of Secure Bloom Filter

Now we make use of HMAC to construct secure Bloom filter that efficiently supports event reporting and message aggregation in wireless sensor networks. Suppose there are c pre-defined events $(e_1, e_2, \ldots, e_c)$, and each event is associated with an event identifier, which is a nonce and may be refreshed by the base station on a regular basis. When node u with identity id_u detects event e_i, whose current event identifier is N_{e_i}, the node creates a secure Bloom filter BF_{u,e_i} as follows.

For simplicity and computational efficiency, we assume that the size of a secure Bloom filter is a power of two, say $m = 2^d$, where d is an integer. We stress that in principal m can be any positive integer. Recall that node u shares a distinct symmetric key K_u with the base station. The node calculates $\mathsf{HMac}(K_u, id_u \| N_{e_i})$ and the output is divided into h equal size pieces of d bits each [2] such that each piece indicates a position on the Bloom filter array. We denote this processing by $\mathsf{BF}_{u,e_i} \leftarrow \mathsf{HMac}(K_u, id_u \| N_{e_i})$. Based on the pseudorandomness assumption of the cryptographic hash function $\mathsf{H}(*)$, it effectively performs as k distinct hash functions with uniformly-distributed output, and all theoretical analysis on the classical Bloom filter still holds for the secure Bloom filter.

4.2.3 Selection of Cryptographic Hash Function

For the specific choice of the cryptographic hash function, at present we select SHA-1. Even though collisions have been found in SHA-1 [28], which raises serious concerns on its security, virtually it does not damage the secure Bloom filter, since its security, same as HMAC [26], relies on second preimage resistance of underlying hash function, rather than the more strong requirement of collision resistance. By now, there is no published result of second preimage attacks against SHA-1. Of course, when SHA-3 is finalized in the future, SHA-1 may be replaced by SHA-3 in the protocol for the purpose of higher security level.

4.3 Security Property of Secure Bloom Filter

We now show how to select appropriate parameters to achieve a desired security level of a secure Bloom filter. The following theorem states the analytic formula of the adversary's advantage against a secure Bloom filter.

Theorem 1. *The advantage of an adversary against secure Bloom filter with parameters* (m, h) *is*

$$\mathrm{Adv}(m, h) = m^{-h} \max_{1 \le i \le h} \tau(h, i) \ , \tag{3}$$

[2] If the output block size of HMAC is greater than dh, it can be truncated. When the size is less than dh, we may conduct several rounds of HMAC operations with different salts to obtain sufficient output.

where

$$\tau(h,i) = \begin{cases} i^h - \sum_{j=1}^{i-1} \binom{i}{j} \tau(h,j) & \text{if } i > 1 \\ 1 & \text{if } i = 1 \end{cases} . \tag{4}$$

Proof. Based on the security of HMAC and the pseudorandomness of cryptographic hash function, we derive the advantage of an adversary against secure Bloom filter with parameters (m, h). Without loss of generality, we suppose that adversary $\mathcal{A}$ is required to output a secure Bloom filter $\mathsf{BF}^{\mathcal{A}}$ which should be identical to $\mathsf{BF}_{u,e}$ for node u claiming event N_e.

Let $p_1(i)$ be the probability of the number of bit ones in $\mathsf{BF}_{u,e}$ being i. Obviously $p_1(i)$ is none-zero only if $1 \leq i \leq h$. Based on the pseudorandomness of cryptographic hash function, we have

$$p_1(i) = \frac{\binom{m}{i} \tau(h,i)}{m^h} ,$$

where $\tau(h,i)$ is equal to the output of the following problem: How many distinct words we can get if we use i different letters to construct words of length h while it is required that all i letters are used.

If the underlining HMAC algorithm is secure, then the adversary only can output random $\mathsf{BF}^{\mathcal{A}}$ with the number of bit ones that he selects. When the bit one number of $\mathsf{BF}_{u,e}$ is i, the probability that a random $\mathsf{BF}^{\mathcal{A}}$ with i bits of one equals $\mathsf{BF}_{u,e}$ is

$$p_2(i) = \frac{1}{\binom{m}{i}} .$$

Subsequently, the overall advantage that adversary can achieve is

$$\mathrm{Adv}(m,h) = \max_{1 \leq i \leq h} p_1(i) p_2(i) = m^{-h} \max_{1 \leq i \leq h} \tau(h,i) .$$

Consider $\tau(h,i)$ in the context of the word-counting problem. When $i = 1$, $\tau(h,i) = 1$. If $i > 1$, by recursion, we can get

$$\tau(h,i) = i^h - \sum_{j=1}^{i-1} \binom{i}{j} \tau(h,j) .$$

This completes the proof. □

In practice, h is relatively small, thus we just explore all values of $\tau(h,i)$ for $1 \leq i \leq h$ by Equation (4), and then retrieve $\max_{1 \leq i \leq h} \tau(h,i)$. Subsequently, $\mathrm{Adv}(m,h)$ can be computed by Equation (3), or we can determine the minimal m satisfying a designated security level. Intuitively, for 80-bit security, the selections of (m,h) should satisfy $\mathrm{Adv}(m,h) \leq 2^{-80}$. To

Table 1: Lookup table for secure Bloom filter's parameters selections

h	i for maximal $\tau(h,i)$	maximal $\tau(h,i)$	minimal m	
			for Adv $\leq 2^{-80}$	for Adv $\leq 2^{-100}$
6	5	1,800	35,999	362,838
7	5	16,800	11,066	80,176
8	6	191,520	4,684	26,495
9	7	2,328,480	2,417	11,278
10	8	30,240,000	1,434	5,733
11	8	4.79E+08	952	3,356
12	9	8.08E+09	680	2,159
13	10	1.43E+11	514	1,492
14	10	2.73E+12	406	1,093
15	11	5.91E+13	334	842
16	12	1.32E+15	282	671

facilitate parameter selections, we calculate and form Table 1 as secure Bloom filter parameters lookup table for $h = 6, 7, \ldots, 16$.

It is worth noting that, by applying Stirling's approximation on

$$p_2(h) = \frac{1}{\binom{m}{h}} = \frac{(m-h)!\ h!}{m!} \ ,$$

we can get

$$p_2(h) \approx h! \left(\frac{e}{m}\right)^h \left(1 - \frac{h}{m}\right)^{m-h+0.5} \ ,$$

which may be further approximated as

$$p_2(h) \approx h!\ e^{(h^2-0.5h)/m} m^{-h} \approx h!\ m^{-h} \ ,$$

when $m \gg h$. In other words, even if an adversary is aware that a particular secure Bloom filter has h bits of ones, the probability that he successfully forges it is roughly $h!\ m^{-h}$.

5 Proposed Protocol

5.1 Protocol Description and Analysis

An interesting property of Bloom filter is inherently supporting aggregation by bitwise-ORing Bloom filters of a same kind together to generate one Bloom filter such that the set represented by the output Bloom filter is the union of the sets of input Bloom filters. Let $\vee$ denote the bitwise-OR operation on Bloom filters. This desired aggregation property perfectly applies to secure Bloom filters generated by sensor nodes with different keys, and thus we take advantage of it in the proposed protocol, which consists of the following four stages.

Stage 1: Initialization

To activate sensor nodes detection on events $(e_1, e_2, \ldots, e_c)$, the base station broadcasts to all nodes an authenticated task message, including fresh nonces $(N_{e_1}, N_{e_2}, \ldots, Ne_c)$ as event identifies, along with events specifications if necessary. For dynamic parameter settings, the task message contains the selected values for the parameters.

Stage 2: Event Claiming and Aggregation

When node u detects that event N_{e_i} occurs, it generates a new Bloom filter

$$\mathsf{BF}_{u,e_i} \leftarrow \mathsf{HMac}(K_u, id_u \| N_{e_i}) \ ,$$

which constitutes the raw message by node u as a contributor to claim event N_{e_i}.

The aggregation on messages is straightforward and efficient. Suppose an aggregator node v receives j messages of BF_i^v, where $i = 1, 2, \cdots, j$, some of which might be contributed by node v itself, it calculates an aggregated message by

$$\mathsf{BF}' = \mathsf{BF}_1^v \vee \mathsf{BF}_2^v \vee \cdots \vee \mathsf{BF}_j^v \ ,$$

To maintain a reasonably slow false positive probability of Bloom filter, aggregators cannot aggregate messages without a limitation. As we see now, the way that we use Bloom filter is quite different from most of other applications. As the message carrier to traverse networks, Bloom filter should not use large m for the sake of communication cost. In fact, $m \leq nc$ for most cases. As result, in addition to m and h for Bloom filter settings, our protocol requires a third parameter: bit one percentage upper-bound ρ. Accordingly, aggregator v tries its best to output one or several aggregated messages with respect to the threshold ρ. In that way, there will be no legitimate Bloom filter whose bit one percentage exceeds ρ. Recall that the false positive probability is optimized when the probability of a bit being 0 in a Bloom filter is 1/2 (see Section 4.1), thus threshold ρ is set to be 50% for most circumstances.

After aggregating messages with respect to ρ, aggregator v sends the aggregated message(s) to next node according to the underlying data aggregation organization.

Stage 3: Processing Reports

Algorithm 1 *ProcessReport*(BF_j): A verifier processes a report BF_j

if the bit one percentage of BF_j exceeds ρ **then**
  ALERT: BF_j is corrupted, discard it
  return false
create an empty $\mathsf{BF}_{\text{test}}$ for testing
for each BF_{u,e_i} **do**
  if $\mathsf{BF}_{u,e_i} \wedge \mathsf{BF}_j = \mathsf{BF}_{u,e_i}$ **then**
    assert that node u has claimed event e_i
    $\mathsf{BF}_{\text{test}} \Leftarrow \mathsf{BF}_{\text{test}} \vee \mathsf{BF}_{u,e_i}$
if $\mathsf{BF}_{\text{test}} \neq \mathsf{BF}_j$ **then**
  ALERT: BF_j is corrupted, discard it
  cancel off all previous assertions
  return false
return true

Eventually, the base station receives l reports BF_j, where $j = 1, 2, \cdots, l$. Using its distinct keys shared with nodes, the base station calculates all nc Bloom filters corresponding to node-event pairs beforehand. For each report, the base station conducts Algorithm 1 to proceed it, intended to discover which nodes claimed which events and to detect attacks.

Let $\wedge$ denote bitwise-AND operation, during the processing procedure, if

$$\mathsf{BF}_{u,e_i} \wedge \mathsf{BF}_j = \mathsf{BF}_{u,e_i} \ ,$$

that is, all bit one occurrences in BF_{u,e_i} appear in BF_j, the base station determines that node u has claimed event e_i. Certainly, false positives may occur for this verification as an inherent downside of Bloom filter. We define the *false positive rate* R_{FP} as the ratio of the count of false positives over the node-event pair number (i.e., nc).

In the ideal case where all l reports are independent and have ρ proportion of bit 1, the false positive probability within a report is ρ^h by Equation (1). Suppose nc is much greater than the raw message number, then the number of false positives roughly follows a binomial distribution of parameter $(nc, 1-(1-\rho^h)^l)$, and then its expected value approximates $nc(1-(1-\rho^h)^l)$. Therefore,

$$R_{\text{FP}}^{(\text{ideal})} \approx 1 - (1-\rho^h)^l \approx 1 - e^{-l\rho^h} \ . \tag{5}$$

In reality, most of reports have less than ρ fraction of bit one, and the expected value of false positive number is related to l as bigger l may indicate more number of raw messages. Therefore, $R_{\text{FP}}^{(\text{ideal})}$ is an upper bound of R_{FP}.

If compromised nodes misleadingly claim events with their own keys, clearly there are no effective methods to prevent this attack except for designing and conducting complicated and expensive detection protocols. We are not concerned about that issue; instead the proposed protocol is supported to detect message manipulation attack. The steps related to $\mathsf{BF}_{\text{test}}$ in Algorithm 1 carry out that mission.

Recall that by selecting proper (m, h), it is infeasible for the adversary to construct a legitimate secure Bloom filter associated with integrity nodes. During transmission, messages may be maliciously altered by adding or deleting some bits. To detect this kind of attack, actually we only need to consider the case that some bits in reports are *randomly* flipped by the adversary. We stress that the base station is solely responsible and capable of processing messages and those effective manipulations shall eventually reflect on reports; if the adversary replaces a message by another message, in fact it is an interception rather than manipulation, and the countermeasure will be addressed later. If the adversary adds a valid message to an unrelated report, that is not a manipulation attack either, and merely it slightly increase the false positive rate, which is still bounded by Equation (5).

We evaluate the probability P_{det} of the base station detecting that a random bit zero in a report is flipped to bit one, which might be the manipulation case that is detected with least probability. Using the previous ideal case of studying R_{FP}, we can get

$$\begin{aligned} P_{\text{det}}^{(\text{ideal})} &\approx \left(1 - \rho^h \left(1 - \left(1 - \frac{1}{\rho m}\right)^h\right)\right)^{nc} \\ &\approx \left(1 - \rho^h + \rho^h e^{-h/\rho m}\right)^{nc} \quad . \end{aligned} \tag{6}$$

Stage 4: Verification by All Nodes

To detect whether malicious nodes intercepted messages related to intact nodes, the base station broadcasts those l reports throughout the whole network. In addition, in order to prevent the message replay attack, the base station should notify all sensor nodes to update event identifiers with new nonces. Surely this stage can be twisted with the initialization stage to add/update/delete events. All those messages are still authenticated by the base station's broadcast authentication scheme.

As long as an intact node is not isolated by compromised nodes from the base station, the node can receive those reports and verity the authenticity of messages. Therefore, the node that claimed the event can determine whether its report is included in the final results. If not, the node notifies this abnormality to the base station in any available secure, uni-cast means by its node key. In addition, if nodes find that there are some false positives related to them, they may report to the base station. If the base station determines that the number of missing raw messages exceeds a reasonable threshold, it may conclude that there exists a message interception attack in the network and further security mechanisms can be conducted.

5.2 Discussions

5.2.1 Compression on Bloom Filter

The entropies of raw messages are pretty low—most of bits are zero because there is only one element in a set that the a raw message Bloom filter represents. As a natural way to reduce the transmission sizes of messages, this kind of Bloom filters should be compressed in our protocol. The problem of compressed Bloom filter has been researched by Mitzenmacher [29], who demonstrates that a better compression rate can be achieved by utilizing smaller h and bigger m while maintaining the same level of false positive probability.

However, that rationale does not fit the case of our protocol. First, the selection of h matters to the security level of secure Bloom filter. Second, there are many Bloom filters with varying entropies during the protocol procedure, one pre-determined parameter set hardly can provide overall desired transmission savings. Therefore, we does not make use of the results in [29]. Instead, a succinct compression mechanism is employed in our protocol: one additional bit b attached to a Bloom filter indicates that the filter is normally stored as before when $b = 0$, and if $b = 1$, then the filter is expressed as a list of all bit one positions. Since the number of bit ones is dynamical, such a list should contain a number-indication segment of bit length $w = \lceil \log_2(m/d) \rceil$, where $m = 2^d$. Before transmitting a Bloom filter, a node chooses the coding which leads to smaller transmission size. Specifically, the list expression is used only if the number of bit ones is less than $(m - w)/d$.

5.2.2 Parameters, Computation and Storage Costs

For sensor networks of size n varying from hundreds to thousands, and to provide 80-bit security level, we recommend $m = 1024, h = 11$ according to Table 1. Because we use SHA-1, whose output size is 160 bit, performing one HMAC function that consists of two SHA-1 calculations would suffice for nodes to construct one raw message. This is the major computation burden on nodes; other calculations such as bit one percentage checking and aggregation are simple bitwise operations and are relatively trivial. We assume that the key length is 160-bit, so the primary storage cost of our protocol is merely 20 bytes. Therefore, our protocol is computation- and storage-efficient.

6 Simulations

To verify our theoretical analysis and evaluate the performance, we implement the proposed data aggregation protocol and run simulations. To facility fast simulation deployment and take advantage of comprehensive result collection tools and inherent statistical support, our simulations are build on the OMNeT++ framework [30], thought we implement secure Bloom filter as a generic class, which is platform-independent.

6.1 Simulation Design

6.1.1 Network Scenarios

We run simulations in two network scenarios. The first one is an abstract hierarchical network modeled as a *Tree* rooted on the base station, which fairly emulates many wireless sensor networks. We generate a tree topology in a way that the number of a non-leaf node's children is uniformly selected from four, five, and six. In our experiments, the network size for the Tree Topology increases from 200 to 2000 at a step of 200.

The other one is *Unit-Disk Graph*, in which nodes are uniformly deployed in a 500×500 square and nodes follow the standard unit-disk bidirectional communication model. We run simulations under the Unit-Disk Graph for network sizes ranging from 500 to 5000 with a space of 500 between. In order to imitate a dense sensor network and maintain consistence, for networks of different sizes, we adjust the node communication range such that the average node degree keeps the approximate 20. This kind of wireless sensor network simulation scenario models random deployed sensor networks and has been widely used in many literatures, such as [31].

By standard software engineering methodologies, the implemented layer of the data aggregation protocol is encapsulated and separated from underling network topologies. With a little effort on topology implement, irregular surveillance areas and other network scenarios, such as random graph and random tree, can be seamlessly introduced to run simulations.

6.1.2 Mechanisms of Claiming Events

On each network scenario, we specify two different workloads to activate nodes claiming events, one for mild work burden, and the other for relatively heavy turnover. For the Tree Topology, in each round of protocol procedure, 20% and 50% of sensor nodes, respectively for two levels of workload, are randomly chosen as contributors to claim an event. For the Unit-Disk Graph, we designate that nodes are capable of detecting events taking place no farther than a distance of 100, and at the beginning of each round, an event occurs at two and six positions (respectively for two workloads and analogue to 20% and 50% contributors settings in the Tree topology) that are uniformly selected in the square.

6.1.3 Simulation Arrangement

Overall, there are four test cases: Tree Topology with mild workload, Tree Topology with heavy workload, Unit-Disk Graph with mild workload, and Unit-Disk Graph with heavy workload. Each case is associated with ten different network sizes, requiring 40 experiments. In order to obtain relatively fair and comparable results, ten random networks in accordance with the parameter settings are constructed for each experiment, each of whose simulation executions is quoted as a run; and in each run, we perform 100 rounds of protocol processing

with new events and randomly selected contributors/event-locations. For the purpose of communication cost comparison, in addition to our proposed protocol, the native scheme described in Section 3.4 is also conducted under the identical conditions. We assume that the message length is 128-bit in the native scheme. Since the last stage is needed for both our protocol and the native scheme, the experiment results on communication costs do not consider that stage. To evaluate the protocol's detection probability P_{det}, we simulate the manipulation attack by randomly flipping a bit zero of a report and then measure whether the base station subsequently detects that the report is illegitimate. For each report, this procedure is repeated by twenty times. The final runs outputs are averaged to create the corresponding experiment results.

6.2 Experimental Results

In figures of this paper, "Proposal", "Native", "(M)", and "(H)" respectively stand for our proposed data aggregation protocol, the native scheme, mild workload, and heavy workload.

6.2.1 Communication Performance

The protocols communication performance along with the comparison is depicted in Figure 1. First, for general sensor network protocols, the communication cost is often represented by average transmission bits per node, which are displayed in Figure 1(a) for Tree Topology and in Figure 1(d) for Unit-Disk Graph. Second, for a data aggregation scheme, the energy consumption of aggregators and the balance among them are critical performance metrics; so we also measure the means (Figure 1(b) for Tree Topology and Figure 1(e) for Unit-Disk Graph) and the standard deviations (Figure 1(c) for Tree Topology and Figure 1(f) for Unit-Disk Graph) of bits sent by aggregators (intermediate nodes for the native scheme).

The simulation results in Figure 1 clearly demonstrate that for all four test cases, our protocol remarkably outweighs the native scheme on all three metrics; not only the protocol reduces the overall energy consumption and reserves previous energy of aggregators, but also it achieves splendid balance among aggregators, which is highly desired for data aggregation scheme. Moreover, as those measurements for our protocol gently grow with n, it proves that the proposed protocol scales to network size well, which is quite appreciated for large-size sensor networks. In fact, when the network size is bigger and/or the workload becomes heavier, the proposed protocol saves much more energy and better balance than the native scheme. On the other hand, the performance improvements for Unit-Dist Graph is generally better than those for Tree Topology, in terms of ratios of same measurements of the proposed protocol over the native scheme. This signifies that the proposed protocol can be reliably used in randomly deployed sensor networks.

It is worth mentioning that the simulations results of average bits sent by aggregator and standard deviations imply that the proposed protocol provides the most energy reduction for

those nodes that are neighbors of the base station and suffer from the heaviest transmission payload, which is supported by our experimental records on this metric. In a word, the proposal is a qualified data aggregation protocol that supplies the desired energy saving for the whole network and more for those critical nodes in need.

6.2.2 Reliability and Security

The simulation results regarding the protocol's reliability and security are illuminated in Figure 2. Specifically, Figure 2(a) and Fig 2(d) depict the protocol false positive rates; Figure 2(b) and Fig 2(e) show the average report numbers; Figure 2(c) and Figure 2(f) exhibit the detection probabilities.

The false positive rates of our protocol are fairly small, compared to typical Bloom filter applications. Even for the most extremely case (Unit-Disk Graph, network size 5000, heavy workload), the false positive rate is still less than 0.5%. For the intended application scenarios, those false positive rates are certainly acceptable. On the other hand, if the base station is aware of all nodes positions, it can easily overcome such small noises within reports and eliminates most of false positives. As the security measurement, the protocol's detection probability in any simulation case is greater than 98%, when a random bit zero in a report is flipped. With such high detection likelihood, any reasonable adversary will not attempt to manipulate reports. In addition, according to Algorithm 1, the detection procedure does not require substantial computation and storage overheads besides indispensable ones for membership checking on a secure Bloom filter. In a word, the proposal is efficient, reliable, and secure.

By applying the experimental results of report number l to Equations (5) and (6), we verify that the previous theoretical analysis on R_{FP} and and P_{det} holds as false positive rates and detection probabilities are indeed bounded on Equations (5) and (6) respectively.

7 Conclusion

In this paper, we propose a practical secure data aggregation protocol for wireless sensor network based on Bloom filter, and conduct thorough theoretical analysis on the related topics. The extensive simulation results sufficiently demonstrate that the proposal presents remarkable performance in terms of communication cost, energy consumption balance, reliability, and security.

References

[1] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next Century Challenges: Scalable Coordination in Sensor Networks," in *Proceedings of the 5th ACM/IEEE Inter-*

national Conference on Mobile Computing and Networking. Seattle: IEEE Computer Society, 1999, pp. 263–270.

[2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, 2002.

[3] R. Rajagopalan and P. K. Varshney, "Data-aggregation techniques in sensor networks: a survey," *IEEE Communications Surveys & Tutorials*, vol. 8, no. 4, pp. 48–63, 2006.

[4] E. Fasolo, M. Rossi, J. Widmer, and M. Zorzi, "In-network aggregation techniques for wireless sensor networks: a survey," *IEEE Wireless Communications*, vol. 14, no. 2, pp. 70–87, 2007.

[5] P. Baronti, P. Pillai, V. W. C. Chook, S. Chessa, A. Gotta, and Y. F. Hu, "Wireless sensor networks: A survey on the state of the art and the 802.15.4 and ZigBee standards," *Computer Communications*, vol. 30, no. 7, pp. 1655–1695, 2007.

[6] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[7] D. Westhoff, J. Girao, and M. Acharya, "Concealed Data Aggregation for Reverse Multicast Traffic in Sensor Networks: Encryption, Key Distribution, and Routing Adaptation," *IEEE Transactions on Mobile Computing*, vol. 5, no. 10, pp. 1417–1431, 2006.

[8] C. Castelluccia, A. C. F. Chan, E. Mykletun, and G. Tsudik, "Efficient and provably secure aggregation of encrypted data in wireless sensor networks," *ACM Trans. Sen. Netw.*, vol. 5, no. 3, pp. 1–36, 2009.

[9] P. Steffen, W. Dirk, and C. Claude, "A Survey on the Encryption of Convergecast Traffic with In-Network Processing," *IEEE Transactions on Dependable and Secure Computing*, vol. 7, pp. 20–34, 2010.

[10] C. Fontaine and F. Galand, "A survey of homomorphic encryption for nonspecialists," *EURASIP Journal on Information Security*, vol. 2007, no. 1, pp. 1–10, 2007.

[11] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the 41st annual ACM symposium on Theory of computing*. Bethesda, MD, USA: ACM, 2009, pp. 169–178.

[12] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully Homomorphic Encryption over the Integers," in *Advances in Cryptology - EUROCRYPT 2010*. LNCS 6110, 2010.

[13] Z. Li and G. Gong, "Data Aggregation Integrity Based on Homomorphic Primitives in Sensor Networks," in *Proceedings of 9th International Conference on Ad Hoc Networks and Wireless (ADHOC-NOW 2010)*. LNCS 6288, 2010, pp. 149–162.

[14] L. Hu and D. Evans, "Secure aggregation for wireless networks," in *Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT'03 Workshops)*, 2003, pp. 384 – 391.

[15] P. Jadia and A. Mathuria, "Efficient Secure Aggregation in Sensor Networks," in *High Performance Computing (HiPC 2004)*. LNCS 3296, 2004, pp. 40–49.

[16] Y. Yang, X. Wang, S. Zhu, and G. Cao, "A Secure Hop-by-Hop Data Aggregation Protocol for Sensor Networks," in *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc'06)*, 2006, pp. 356 – 367.

[17] B. Przydatek, D. Song, and A. Perrig, "SIA: Secure Information Aggregation in Sensor Networks," in *Proceedings of the first international conference on Embedded networked sensor systems (SenSys'03)*, Los Angeles, California, USA, 2003, pp. 255 – 265.

[18] H. Chan, A. Perrig, and D. Song, "Secure hierarchical in-network aggregation in sensor networks," in *Proceedings of the 13th ACM conference on Computer and communications security*. Alexandria, Virginia, USA: ACM, 2006, pp. 278–287.

[19] K. B. Frikken and J. A. Dougherty I. V., "An efficient integrity-preserving scheme for hierarchical sensor aggregation," in *Proceedings of the first ACM conference on Wireless network security (WiSec'08)*. Alexandria, VA, USA: ACM, 2008, pp. 68–76.

[20] H. Chan and A. Perrig, "Efficient security primitives derived from a secure aggregation algorithm," in *Proceedings of the 15th ACM conference on Computer and communications security (CCS'08)*. Alexandria, Virginia, USA: ACM, 2008.

[21] Y. Liu, C. Li, Y. He, J. Wu, and Z. Xiong, "A Perimeter Intrusion Detection System Using Dual-Mode Wireless Sensor Networks," in *Second International Conference on Communications and Networking in China (CHINACOM '07)*, 2007, pp. 861–865.

[22] A. Perrig, R. Szewczyk, V. W. D. Culler, and J. D. Tygar, "SPINS: Security protocols for sensor networks," in *Proceedings of the Annual International Conference on Mobile Computing and Networking (MOBICOM)*. Rome Italy: IEEE, 2001, pp. 189–199.

[23] J. Katz and A. Lindell, "Aggregate Message Authentication Codes," in *Topics in Cryptology - CT-RSA 2008*. LNCS 4964, 2008, pp. 155–169.

[24] L. Fan, P. Cao, J. Almeida, and A. Z. Broder, "Summary cache: a scalable wide-area web cache sharing protocol," *IEEE/ACM Transactions on Networking*, vol. 8, no. 3, pp. 281–293, 2000.

[25] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: a distributed storage system for structured data," in *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation*, vol. 7. Seattle, WA: USENIX Association, 2006, pp. 15–15.

[26] M. Bellare, R. Canetti, and H. Krawczyk, "Keying Hash Functions for Message Authentication," in *Advances in Cryptology - CRYPTO'96*. LNCS 1109, 1996, pp. 1–15.

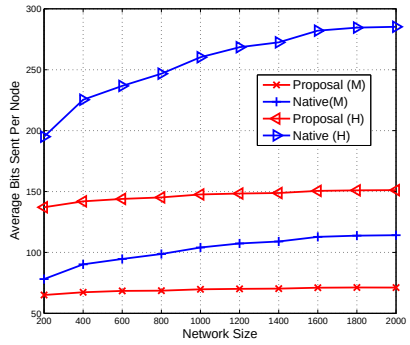
[27] H. Krawczyk, M. Bellare, and R. Canetti, "RFC 2104 - HMAC: Keyed-Hashing for Message Authentication," IETF, 1997.

[28] X. Wang, Y. L. Yin, and H. Yu, "Finding Collisions in the Full SHA-1," in *Advances in Cryptology - CRYPTO 2005*. LNCS 3621, 2005, pp. 17–36.

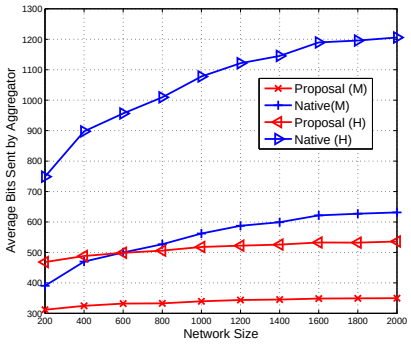
[29] M. Mitzenmacher, "Compressed bloom filters," *IEEE/ACM Transactions on Networking*, vol. 10, no. 5, pp. 604–612, 2002.

[30] A. Varga and R. Hornig, "An overview of the OMNeT++ simulation environment," in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*. Marseille, France: ICST, 2008, pp. 1–10.

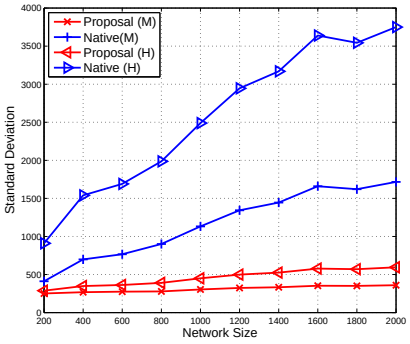
[31] B. Parno, A. Perrig, and V. Gligor, "Distributed Detection of Node Replication Attacks in Sensor Networks," in *Proceedings of the IEEE Symposium on Security and Privacy*, 2005, pp. 49–63.



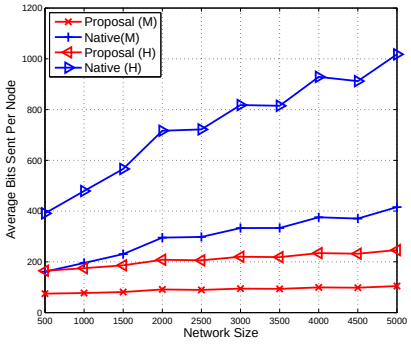
(a) Average Communication Cost (Tree)



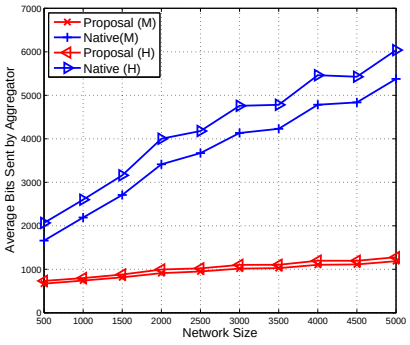
(b) Aggregators Communication Cost, Mean (Tree)



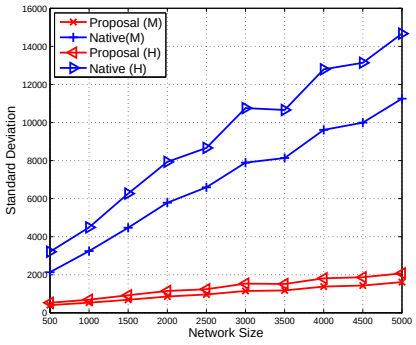
(c) Aggregators Communication Cost, Standard Deviation (Tree)



(d) Average Communication Cost (Unit-Disk)

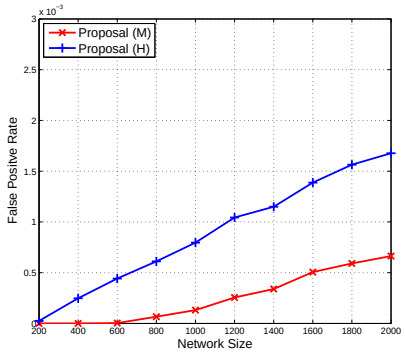


(e) Aggregators' Communication Cost, Mean (Unit-Disk)

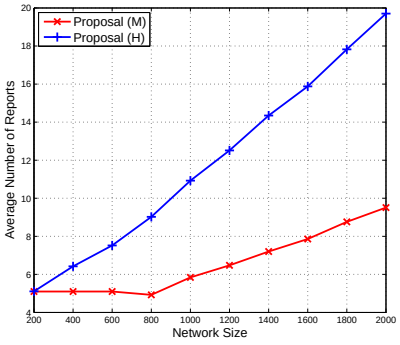


(f) Aggregators' Communication Cost, Standard Deviation (Unit-Disk)

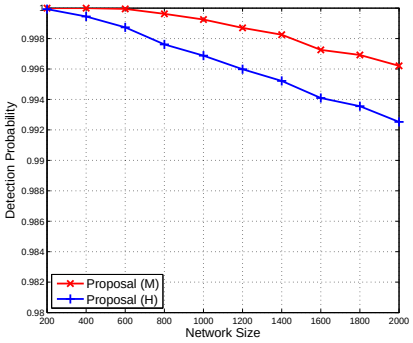
Figure 1: Simulation Results on Protocols Communication Performance



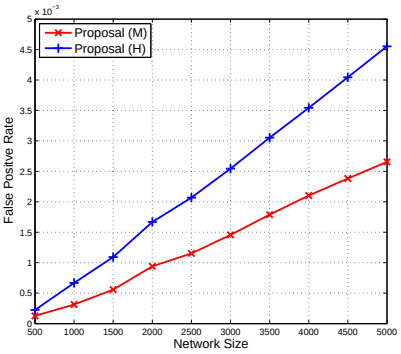
(a) False Positive Rate (Tree)



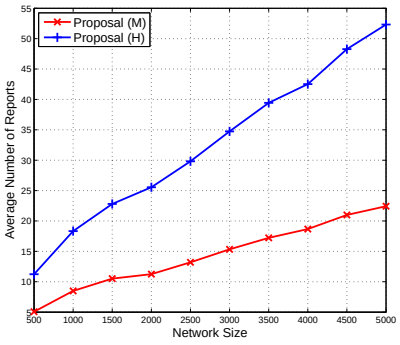
(b) Report Number (Tree)



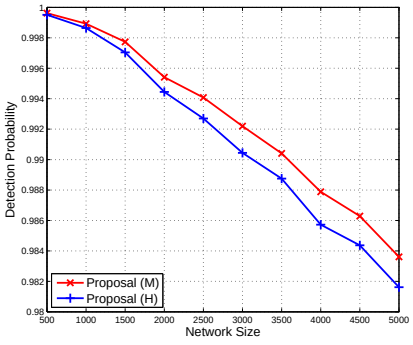
(c) Detection Probability (Tree)



(d) False Positive Rate (Unit-Disk)



(e) Report Number (Unit-Disk)



(f) Detection Probability (Unit-Disk)

Figure 2: Simulation Results Related to Protocol's Reliability and Security